In [41]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import sklearn
```

In [42]:
```
data = pd.read_csv('car-sales-extended-missing-data.csv')
data.head()
```

Out[42]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Honda	White	35431.0	4.0	15323.0
1	BMW	Blue	192714.0	5.0	19943.0
2	Honda	White	84714.0	4.0	28343.0
3	Toyota	White	154365.0	4.0	13434.0
4	Nissan	Blue	181577.0	3.0	14043.0

In [43]:
```
for label, content in data.items():
    if pd.api.types.is_numeric_dtype(content):
        print(label)
```

```
Odometer (KM)
Doors
Price
```

In [44]:
```
for label, content in data.items():
    if pd.api.types.is_numeric_dtype(content):
        if pd.isnull(content).sum():
            print(label)
```

```
Odometer (KM)
Doors
Price
```

In [45]:
```
for label, content in data.items():
    if pd.api.types.is_numeric_dtype(content):
        data[label+'_is_missing'] = pd.isnull(content)
        data[label] = content.fillna(content.median())
```

In [46]:
```
data.isna().sum()
```

Out[46]:
```
Make                        49
Colour                      50
Odometer (KM)                0
Doors                        0
Price                        0
Odometer (KM)_is_missing     0
Doors_is_missing             0
Price_is_missing             0
dtype: int64
```

In [47]:
```
def convert_columns_to_categorical (df, column_list):
    for column in column_list:
        if pd.api.types.is_string_dtype(df[column]):
            df[column] = df[column].astype('category')
        elif pd.api.types.is_numeric_dtype(df[column]):
            df[column] = pd.Categorical(df[column])
        else:
            print(f'Column {column} is not a string or numeric type.')

df = pd.DataFrame(data)
convert_columns_to_categorical(df, ['Make', 'Colour', 'Doors'])
```

In [48]:
```
for column in ['Make', 'Colour', 'Doors']:
    df[column] = df[column].cat.codes

```

In [49]:
```
1 df
```

Out[49]:

	Make	Colour	Odometer (KM)	Doors	Price	Odometer (KM)_is_missing	Doors_is_missing	Price_is_missing
0	1	4	35431.0	1	15323.0	False	False	False
1	0	1	192714.0	2	19943.0	False	False	False
2	1	4	84714.0	1	28343.0	False	False	False
3	3	4	154365.0	1	13434.0	False	False	False
4	2	1	181577.0	0	14043.0	False	False	False
...	...	...	...	...	...	...	...	...
995	3	0	35820.0	1	32042.0	False	False	False
996	-1	4	155144.0	0	5716.0	False	False	False
997	2	1	66604.0	1	31570.0	False	False	False
998	1	4	215883.0	1	4001.0	False	False	False
999	3	1	248360.0	1	12732.0	False	False	False

1000 rows × 8 columns

In [50]:
```
1 df.isna().sum()
```

Out[50]:
```
Make                        0
Colour                      0
Odometer (KM)               0
Doors                       0
Price                       0
Odometer (KM)_is_missing    0
Doors_is_missing            0
Price_is_missing            0
dtype: int64
```

In [51]:
```
1 x = df.drop('Price', axis = 1)
2 y = df['Price']
```

In [52]:
```
train = x
train
```

Out[52]:

	Make	Colour	Odometer (KM)	Doors	Odometer (KM)_is_missing	Doors_is_missing	Price_is_missing
0	1	4	35431.0	1	False	False	False
1	0	1	192714.0	2	False	False	False
2	1	4	84714.0	1	False	False	False
3	3	4	154365.0	1	False	False	False
4	2	1	181577.0	0	False	False	False
...	...	...	...	...	...	...	...
995	3	0	35820.0	1	False	False	False
996	-1	4	155144.0	0	False	False	False
997	2	1	66604.0	1	False	False	False
998	1	4	215883.0	1	False	False	False
999	3	1	248360.0	1	False	False	False

1000 rows × 7 columns

In [53]:
```
test = y
test
```

Out[53]:
```
0      15323.0
1      19943.0
2      28343.0
3      13434.0
4      14043.0
        ...
995    32042.0
996     5716.0
997    31570.0
998     4001.0
999    12732.0
Name: Price, Length: 1000, dtype: float64
```

In [54]:
```
1 test.to_csv('test_data.csv', index = False)
```

In [55]:
```
1 train
```

Out[55]:

	Make	Colour	Odometer (KM)	Doors	Odometer (KM)_is_missing	Doors_is_missing	Price_is_missing
0	1	4	35431.0	1	False	False	False
1	0	1	192714.0	2	False	False	False
2	1	4	84714.0	1	False	False	False
3	3	4	154365.0	1	False	False	False
4	2	1	181577.0	0	False	False	False
...	...	...	...	...	...	...	...
995	3	0	35820.0	1	False	False	False
996	-1	4	155144.0	0	False	False	False
997	2	1	66604.0	1	False	False	False
998	1	4	215883.0	1	False	False	False
999	3	1	248360.0	1	False	False	False

1000 rows × 7 columns

In [76]:
```
1 from sklearn.ensemble import RandomForestRegressor
2 from sklearn.ensemble import RandomForestRegressor
3
4 %time
5 model = RandomForestRegressor()
6 model.fit(df.drop('Odometer (KM)', axis = 1), df['Odometer (KM)'])
7
8
```

```
Wall time: 0 ns
```

Out[76]:
```
RandomForestRegressor()
```

In [77]:
```
model.score(df.drop('Odometer (KM)', axis = 1), df['Odometer (KM)'])
```

Out[77]:
```
0.8413747570506006
```

In [78]:
```
df_test = pd.read_csv('test_data.csv')
df_test
```

Out[78]:

	Price
0	15323.0
1	19943.0
2	28343.0
3	13434.0
4	14043.0
...	...
995	32042.0
996	5716.0
997	31570.0
998	4001.0
999	12732.0

1000 rows × 1 columns

In [79]:
```
1 test_preds = model.predict(df_test)
```

```
C:\Users\USER\anaconda3\lib\site-packages\sklearn\base.py:493: FutureWarning: The feature names should m
atch those that were passed during fit. Starting version 1.2, an error will be raised.
Feature names seen at fit time, yet now missing:
- Colour
- Doors
- Doors_is_missing
- Make
- Odometer (KM)_is_missing
- ...

  warnings.warn(message, FutureWarning)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_7504\1359886214.py in <module>
----> 1 test_preds = model.predict(df_test)

~\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in predict(self, X)
    969         check_is_fitted(self)
    970         # Check data
--> 971         X = self._validate_X_predict(X)
    972
    973         # Assign chunk of trees to jobs

~\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in _validate_X_predict(self, X)
    577         Validate X whenever one tries to predict, apply, predict_proba."""
    578         check_is_fitted(self)
--> 579         X = self._validate_data(X, dtype=DTYPE, accept_sparse="csr", reset=False)
    580         if issparse(X) and (X.indices.dtype != np.intc or X.indptr.dtype != np.intc):
    581             raise ValueError("No support for np.int64 index based sparse matrices")

~\anaconda3\lib\site-packages\sklearn\base.py in _validate_data(self, X, y, reset, validate_separately,
**check_params)
    583
    584         if not no_val_X and check_params.get("ensure_2d", True):
--> 585             self._check_n_features(X, reset=reset)
    586
    587         return out

~\anaconda3\lib\site-packages\sklearn\base.py in _check_n_features(self, X, reset)
    398
    399         if n_features != self.n_features_in_:
--> 400             raise ValueError(
    401                 f"X has {n_features} features, but {self.__class__.__name__} "
    402                 f"is expecting {self.n_features_in_} features as input."

ValueError: X has 1 features, but RandomForestRegressor is expecting 7 features as input.
```

In [ ]:
```
1 train
```

In [ ]:
```
df_test
```

In [ ]:
```
df_filled = pd.DataFrame(df_test)
df_test['Make'] = False
df_test['Colour'] = False
df_test['Odometer (KM)'] = False
df_test['Odometer (KM)_is_missing'] = False
df_test['Doors_is_missing'] = False
df_test['Price_is_missing'] = False

```

In [80]:
```
df_filled
```

Out[80]:

	Price	Colour	Doors	Doors_is_missing	Make	Odometer (KM)_is_missing	Price_is_missing
0	15323.0	False	False	False	False	False	False
1	19943.0	False	False	False	False	False	False
2	28343.0	False	False	False	False	False	False
3	13434.0	False	False	False	False	False	False
4	14043.0	False	False	False	False	False	False
...	...	...	...	...	...	...	...
995	32042.0	False	False	False	False	False	False
996	5716.0	False	False	False	False	False	False
997	31570.0	False	False	False	False	False	False
998	4001.0	False	False	False	False	False	False
999	12732.0	False	False	False	False	False	False

1000 rows × 7 columns

In [81]:
```
1 test_preds = model.predict(df_filled)
```

```
C:\Users\USER\anaconda3\lib\site-packages\sklearn\base.py:493: FutureWarning: The feature names should m
atch those that were passed during fit. Starting version 1.2, an error will be raised.
Feature names must be in the same order as they were in fit.

  warnings.warn(message, FutureWarning)
```

The feature names should match those that were passed during fit. Starting version 1.2, an error will be raised. Feature names must be in the same order as they were in fit.

warnings.warn(message, FutureWarning)

In [ ]:
```
1
```